The "Buffered Interface"

We introduced a new class of API routines to the 1.3.0 release. These "buffered put APIs" (eg. ncmpi_bput_vara_float) make a copy of the user's buffer internally, so the user's buffer can be reused when the call returns. Their usage are similar to the iput APIs.

The usage of bput APIs is very similar to iput, except the followings.

1. users must tell PnetCDF the size of buffer to be used by PnetCDF internally (attach and detach calls).
2. once a bput API returns, user's buffer can be reused or freed (because the write data has been copied to the internal buffer.)

The internal buffer is per file (as the attach API requires an ncid argument.) It can be used to aggregate requests to multiple variables defined in the file.

The buffer will not be flushed automatically and all file I/O happens in wait_all. If the attached buffer ran out of space, NC_EINSUFFBUF error code (non-fatal) will return. It can be used to determine to call wait API, as described above. However, an automatic flushing would require an MPI independent I/O, again meaning a poor performance. So, users are recommended to make sure the buffer size is sufficient large. In addition, if you rely on PnetCDF to do type conversion between the I/O buffer and file data in two data types of different size (e.g. I/O buffer is in short and NC file variable is in 4-byte int), you must calculate the size of attach buffer using the larger type. Once a call to wait/wait_all returns, the internal buffer usage is reduced by those completely requests.

Here is an example code fragment for calling buffered APIs:

```
ncmpi_buffer_attach(ncid, bufsize);
ncmpi_bput_vara_float(ncid, varid0, start0, count0, buf0, &req[0]);
ncmpi_bput_vara_float(ncid, varid0, start1, count1, buf1, &req[1]);
ncmpi_bput_vara_int64(ncid, varid1, start2, count2, buf2, &req[2]);
ncmpi_wait_all(ncid, 3, req, status);
ncmpi_buffer_detach(ncid);
```

Check the current usage of the buffer

A new inquiry API is added to check the usage of the buffer. It can be useful to know how much space is left for more buffered APIs calls. This inquiry API can be called in either collective or independent data mode.

```
int ncmpi_inq_buffer_usage(int ncid, MPI_Offset *usage);
```

- "usage" will be returned with the current buffer usage in bytes.
- Error codes may be invalid ncid (NC_EBADID) or no attached buffer found (NC_ENULLABUF).

Possible Future Directions

When the buffer fills up, the buffered APIs return NC_EINSUFFBUF and that request is skipped. It might be possible to implement automatic flushing of the buffer, but as this process would be un-coordinated among processors, it would have to be done independently and so would probably not perform very well. Perhaps there is a clever idea we have yet to come across.

Examples

Fortran: http://trac.mcs.anl.gov/projects/parallel-netcdf/browser/trunk/examples/tutorial/pnetcdf-write-bufferedf.F

C: http://trac.mcs.anl.gov/projects/parallel-netcdf/browser/trunk/examples/tutorial/pnetcdf-write-buffered.c

Availability

- The buffered interface was introduced in 1.3.0. As this is a new interface and still subject to experimentation, we recommend tracking PnetCDF's SVN trunk for the latest implementation.
- The buffer usage inquiry API ncmpi_inq_buffer_usage/nfmpi_inq_buffer_usage is added in 1.3.1